IJCIT (Indonesian Journal on Computer and Information Technology) 6 (1) (2021) 20-24
Diterima 1 Desember 2020; Revisi 2 Februari 2020; Disetujui 17 Februari 2021

IJCIT

(Indonesian Journal on Computer and Information Technology)

Journal Homepage: http://ejournal.bsi.ac.id/ejurnal/index.php/ijcit

Test-Driven Development pada Pengembangan Aplikasi Android untuk Memantau COVID-19

Federick Jonathan[1], Magdalena A. Ineke Pakereng[2]

Teknik Informatika, Universitas Kristen Satya Wacana
Salatiga, Indonesia
e-mail: federickjonathan22@gmail.com[1], ineke.pakereng@uksw.edu[2]

A B S T R A K

Dalam pengembangan perangkat lunak, terdapat banyak teknik dan pendekatan yang digunakan untuk menghasilkan perangkat lunak yang handal. Kualitas perangkat lunak sangat bergantung pada pengujian perangkat lunak. Namun tidak semua pengembang peduli dengan tahapan pengujian pada sebuah perangkat lunak. Penelitian ini bertujuan untuk mengetahui pengaruh dari menerapkan proses pengujian dalam mengembangkan perangkat lunak dengan menggunakan metode TDD. Pada Metode TDD, pengembangan perangkat lunak dimulai dengan menulis test case terlebih dahulu lalu kemudian menulis kode. Pada artikel ini, dikembangkan aplikasi mobile dengan menerapkan metode TDD. Perangkat lunak yang dikembangkan adalah berupa sistem informasi mengenai data laporan kasus COVID-19. Data diambil dari Johns Hopkins University The Center of Systems Science and Engineering (JHU CSSE). Hasil penerapan metode TDD menunjukkan bahwa fungsi dan fitur dari perangkat lunak yang dibangun dapat bekerja dan terintegrasi dengan baik antar satu sama lain. Kode yang dihasilkan dari penerapan TDD juga menjadi rapih karena dilakukannya proses refactoring.
Kata Kunci: aplikasi mobile, pengujian, perangkat lunak, t*est driven development*

A B S T R A C T S

In Software Engineering, there are many techniques and approaches that can be used to build a reliable software. The quality of a software relies mostly on the software testing process. However, not many developers are bothered with the testing step of a software. The purpose of this article is to learn the results from implementing a testing process on software developmenty. In TDD, the development is started by writing test case first and then writing code. This article developed a mobile application by applying TDD in the process. The android application that had been developed is an information system about report cases on COVID-19. The cases are coming from Johns Hopkins University The Center of Systems Science and Engineering (JHU CSSE). The result of using TDD in development proves that all functions and features of the developed application are working and integrated well.
Keywords*: coronavirus disease, mobile application, software, test driven development, testing*

1. PENDAHULUAN

Test Driven Development (TDD) adalah salah satu praktik yang umum dalam inti pengembangan metode *Agile* (Khanam & Ahsan, 2017). Metode TDD menuntut pengembang untuk menentukan terlebih dahulu proses bisnis dan desain perangkat lunak yang dibutuhkan, setelah itu baru menulis kode. Ini menjamin stabilitas untuk meningkatkan kepercayaan diri developer dan meraih cakupan tingkatan tinggi pengujian untuk sistem yang longgar dan berhubungan. Ini juga meningkatkan kejelasan mengenai ruang lingkup pengembangan.

TDD digunakan untuk mengembangkan perangkat lunak gabungan antara desain dan

Jurnal ini dapat diakses secara terbuka dan memiliki lisensi CC-BY-SA
(https://creativecommons.org/licenses/by-sa/4.0/) ©2021 by penulis dan IJCIT

pengujian logika program. Oleh karena itu dianggap sebagai penyatuan dari *Test First Development* dimana *unit test* dilakukan terlebih dahulu sebelum *writing code*. *Refactoring* juga terkait dengan proses TDD dan ini berperan penting dalam menstruktur ulang potongan kode. Selanjutnya, tes harus berhasil untuk mengurangi kompleksitas dan meningkatkan pemahaman, pemeliharaan, dan kejelasan kode (Buffardi & Edwards, 2012; Ivo et al., 2018; Moe, 2019; Muhammad Shahid Khan, 2013).

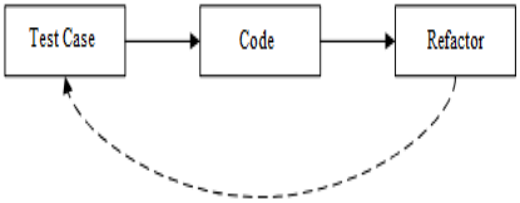
Pengujian perangkat lunak dilakukan dengan tujuan untuk meningkatkan dan menilai keandalan dan menjamin kinerja perangkat lunak agar dalam proses pengembangannya terhindar dari *error* atau kegagalan yang tidak dinginkan. Sistem yang mempunyai *bug* atau *error* tentu saja akan mempengaruhi kinerja perangkat lunak seperti kegagalan dalam pemrosesan data, kegagalan dalam mengirim data dan kegagalan pada hasil yang diinginkan (Bulajic et al., 2012; Rós Aguilar, 2016). Dengan demikian perangkat lunak membutuhkan proses pengujian yang mendetail pada komponen-komponen yang ada sebelum komponen-komponen tersebut diintegrasikan satu sama lain hingga menjadi suatu sistem untuk digunakan (Khanam, 2018).

Coronavirus disease 2019 (COVID-19) adalah penyakit menular yang disebabkan oleh SARS-CoV-2. Penyakit ini pertama kali teridentifikasi pada Desember 2019 di Wuhan dan telah menyebar secara global yang menyebabkan pandemi coronavirus 2019-2020. Seseorang yang terjangkit penyakit ini dapat mengalami demam, kesulitan bernapas, dan batuk kering. Namun, virus ini juga bisa menyebabkan infeksi pernapasan berat, seperti infeksi paru-paru (pneumonia). Penyebaran virus dapat terjadi melalui kontak dekat dan tidak sengaja menghirup atau memegang mulut setelah menyentuh benda yang terkena percikan ludah dari penderita COVID-19. Rutin mencuci tangan dengan sabun atau *hand sanitizer*, menerapkan *physical distancing*, dan tidak menyentuh mata, mulut, dan hidung sebelum mencuci tangan adalah langkah-langkah pencegahan COVID-19 (Brahma, 2020; Hamid, 2020).

2. METODE PENELITIAN

Metode TDD dalam peneliatan ini digunakan untuk menguji API COVID-19. Langkah pertama adalah membuat pengujian sesuai dengan proses bisnis dan desain dari sistem yang diinginkan dengan menyiapkan *mocking* untuk unit yang akan diuji (*test case*). Langkah kedua adalah menulis kode implementasi untuk unit dari *mocking* tersebut hingga lulus uji (*code*). Lalu langkah ketiga, kode implementasi unit yang sudah lulus uji dapat diperbaiki dan dirapihkan agar lebih mudah dipahami dan dikembangkan lebih lanjut (*refactor*), sebagaimana dapat dilihat pada Gambar 1.

Dengan menerapkan metode TDD, proses penggabungan unit akan menjadi lebih mudah karena setiap unit dipastikan bekerja dengan semestinya sebelum diintegrasikan satu sama lain. Manfaat dari menerapkan TDD adalah kemudahan untuk menemukan unit penyebab *bug* atau *error* dalam suatu sistem.



Gambar 1. Alur Kerja TDD

3. HASIL DAN PEMBAHASAN

3.1. Pengujian Unit

Proses pengujian unit dibuat dengan bantuan *package test* dan *package flutter_test* yang sudah disediakan oleh *Flutter SDK*. Dua hal ini memudahkan proses pengujian karena sudah terintegrasi secara langsung dalam *Flutter SDK*. Kode program dapat dilihat di Gambar 2.

Pada Gambar 2 merupakan kode program untuk melakukan pengujian terhadap status *caching* perangkat lunak, dimana tahapannya adalah sebagai berikut. Jika *cache* kosong, maka akan melakukan pengambilan data dari API. Sebaliknya, perangkat akan menampilkan data dari hasil *cache* yang ada. Data yang diambil dari API adalah total terkonfirmasi terjangkit, total kematian dan total sembuh.

Gambar 3 merupakan hasil *running* dari pengujian yang sudah dibuat. Hasil *running* gagal dikarenakan belum ada implementasi kode yang ditulis. Ketika sudah melakukan langkah ini, maka langkah selanjutnya adalah menulis kode untuk membuat pengujian unit yang sudah dibuat berhasil.

```
Run | Debug
test(
  'initialState should return GlobalSummaryInitial() when fromJson returns null',
  () {
    when<dynamic>(storage.read('GlobalSummaryBloc')).thenReturn(null);
    expect(globalSummaryBloc.initialState, GlobalSummaryInitial());
    verify<dynamic>(storage.read('GlobalSummaryBloc')).called(2);
  },
);

Run | Debug
test(
  'initialState should return GlobalSummaryLoaded() when fromJson returns summary',
  () {
    when<dynamic>(storage.read('GlobalSummaryBloc'))
        .thenReturn(json.encode({
      'time': time,
      'summary': summary.toJson(),
    }));
    expect(globalSummaryBloc.initialState,
        GlobalSummaryLoaded(time, summary));
    verify<dynamic>(storage.read('GlobalSummaryBloc')).called(2);
  },
);
```

Gambar 2. Test case Cache Storag

```
Expected: [GlobalSummaryLoaded:GlobalSummaryLoaded]
  Actual: []
   Which: shorter than expected at location [0]

package:test_api                              expect
package:bloc_test/src/bloc_test.dart 124:25   blocTest.<fn>

✕ mapEventToState emits [GlobalSummaryLoaded()] when InitGlobalSummary() is added
Exited (1)
```

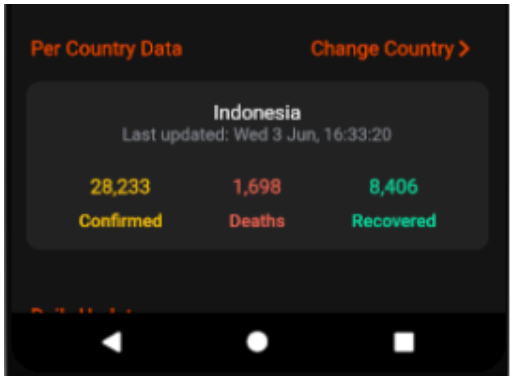
Gambar 3. Hasil Pengujian Gagal Pada Fitur Global Summary

3.2. Implementasi Kode

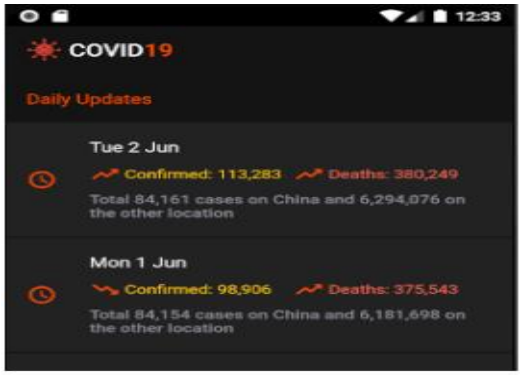
Langkah selanjutnya adalah membuat implementasi kode program untuk berhasil melewati pengujian unit yang sudah dibuat, hasil implementasi dapat dilihat pada Gambar 4, Gambar 5 dan Gambar 6. Pada Gambar 4 merupakan fitur utama COVID-19 Monitor yaitu laporan global kasus-kasus berkaitan dengan COVID-19. Terdapat empat elemen, total terkonfirmasi terjangkit, total kematian, total sembuh dan waktu terakhir pengguna mengecek. Pada Gambar 5 merupakan fitur untuk mengecek kondisi terbaru berdasarkan negara pilihan. Sama halnya dengan *Global Summary*, *Per Country Data* memiliki elemen yang sama ditambah dengan nama negara terpilih. Pada Gambar 6 merupakan fitur yang menampilkan masukan data terbaru harian disertai dengan detail setiap negara dan provinsinya.

Pada tahap selanjutnya, kode yang ditulis harus berhasil melewati pengujian. Hasil pengujian dapat dilihat pada Gambar 7, Gambar 8 dan Gambar 9.

Gambar 4. Global Summary



Gambar 5. Per Country Data



Gambar 6. Daily Updates

```
✓ initialState HydratedBloc initialState should return GlobalSummaryInitial() when fromJson returns null
✓ initialState HydratedBloc initialState should return GlobalSummaryLoaded() when fromJson returns summary
✓ initialState HydratedBloc clear calls delete on storage
✓ mapEventToState emits [GlobalSummaryLoaded()] when RefreshGlobalSummary() is added
✓ mapEventToState emits [GlobalSummaryLoaded()] when InitGlobalSummary() is added
Exited
```

Gambar 7. Pengujian Berhasil Pada Fitur Global Summary

```
✓ initialState HydratedBloc initialState should return PerCountryInitial() when fromJson returns null
✓ initialState HydratedBloc initialState should return PerCountryLoaded() when fromJson returns perCountry
✓ initialState HydratedBloc clear calls delete on storage
✓ mapEventToState emits [PerCountryLoaded()] when ChangePerCountry() is added
Exited
```

Gambar 8. Pengujian Berhasil Pada Fitur Per Country Data

```
✓ initialState HydratedBloc initialState should return DailyUpdateInitial() when fromJson returns null
✓ initialState HydratedBloc initialState should return DailyUpdateLoaded() when fromJson returns dailyUpdate
✓ initialState HydratedBloc clear calls delete on storage
✓ mapEventToState emits [DailyUpdateLoaded()] when RefreshDailyUpdate() is added
✓ mapEventToState emits [DailyUpdateLoaded()] when InitDailyUpdate() is added
Exited
```

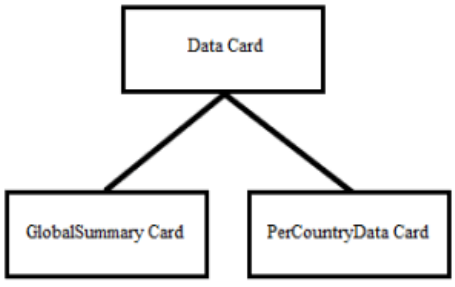
Gambar 9. Pengujian Berhasil Pada Fitur Daily Updates

Pengujian untuk fitur lainnya seperti Country List, Daily Country dan Scrolling Position juga sudah dilakukan, walau tidak dicantumkan dalam artikel ini dikarenakan kemiripan dengan pengujian unit pada fitur-fitur tersebut.

3.3. Refactoring Code

Refactoring code adalah proses menstruktur ulang kode yang sudah diimplementasikan tanpa mengubah proses bisnisnya. Ini bertujuan untuk memproduksi kode yang mudah dibaca, mengurangi kompleksitas, memudahkan pemeliharaan dan tidak mengandung duplikasi.

Pada Gambar 10 menunjukkan bahwa *Data Card* mengandung *template* yang dapat digunakan bersama oleh *GlobalSummary* dan *PerCountryData*. Sehingga dapat meningkatkan *readability* dari potongan kode. Kemudian pada Tabel 1 menunjukkan bahwa hasil pengujian unit dengan menerapkan metode TDD pada tiap unit telah berhasil.



Gambar 10. Refactoring Pada GlobalSummary Card dan PerCountryData Card

Tabel 1. Hasil Pengujian Unit

Unit	Hasil Pengujian
Global Summary	Berhasil
Global Summary Cache	Berhasil
PerCountryData	Berhasil
PerCountryData Cache	Berhasil
CountryList	Berhasil
DailyUpdates	Berhasil
DailyUpdates Cache	Berhasil
DailyCountry	Berhasil
ScrollingPosition	Berhasil
ScrollingPosition Cache	Berhasil

4. KESIMPULAN

Berdasarkan hasil implementasi dan perancangan dengan menerapkan metode TDD, dapat disimpulkan bahwa fitur-fitur dari perangkat lunak mobile COVID-19 Monitor yang dibangun berjalan dengan baik. Dengan menerapkan metode TDD, pengembang dimudahkan dalam menemukan kesalahan dalam proses implementasi kode. Adanya proses *refactoring* dalam metode TDD menghasilkan kode yang lebih rapih setelah berhasil mengimplementasikan kode. Disamping beberapa kemudahan yang ada, penerapan metode TDD pada pengembangan aplikasi mobile ini juga menimbulkan kesulitan selama proses pengembangan. Salah satu kesulitan dalam menerapkan metode TDD adalah pada proses mengubah kebutuhan dan proses bisnis yang diinginkan pengguna ke dalam bentuk *test case*. Terkadang *test case* yang sudah ditulis tidak sesuai atau belum tepat dengan desain dan proses bisnis yang diinginkan, sebab tidak ada indikasi yang menunjukkan bahwa *test case* yang ditulis sudah tepat sesuai proses bisnis.

5. REFERENSI

Brahma, B. (2020). COVID-19 and Oncologists in Indonesia : What can we learn and should do ? *Indonesian Journal of Cancer*, 1–2.

Buffardi, K., & Edwards, S. H. (2012). Impacts of Teaching Test-Driven Development to Novice Programmers. *International Journal of Information and Computer Science IJICS*, *1*(6), 135–143. www.ijics.org

Bulajic, A., Sambasivam, S., & Stojic, R. (2012). Overview of the Test Driven Development Research Projects and Experiments. *Proceedings of the 2012 InSITE Conference*, *2011*, 165–187. https://doi.org/10.28945/1647

Hamid, A. R. A. H. (2020). Social responsibility of medical journal: A concern for covid-19 pandemic. *Medical Journal of Indonesia*, *29*(1), 1–3. https://doi.org/10.13181/mji.ed.204629

Ivo, A. A. S., Guerra, E. M., Porto, S. M., Choma, J., & Quiles, M. G. (2018). An approach for applying Test-Driven Development (TDD) in the development of randomized algorithms. *Journal of Software Engineering Research and Development*, *6*(1). https://doi.org/10.1186/s40411-018-0053-5

Khanam, Z. (2018). Barriers to Refactoring : Issues and Solutions. *International Journal on Future Revolution in Computer Science & Communication Engineering*, *4*, 232–235.

Khanam, Z., & Ahsan, M. N. (2017). Evaluating the effectiveness of test driven development: Advantages and pitfalls. *International Journal of Applied Engineering Research*, *12*(18), 7705–7716.

Moe, M. M. (2019). Comparative Study of Test-Driven Development TDD, Behavior-Driven Development BDD and Acceptance Test–Driven Development ATDD. *International Journal of Trend in Scientific Research and Development*, *Volume-3*(Issue-4), 231–234. https://doi.org/10.31142/ijtsrd23698

Muhammad Shahid Khan, N. K. M. A. K. M. A. J. (2013). Reducing Testing Effort in the Test Driven Development. *Global Journal of Computer Science and Technology*, *13*(7), 0–4.

Rós Aguilar, R. (2016). *Using Test-Driven Development to Improve Software Development Practices*.